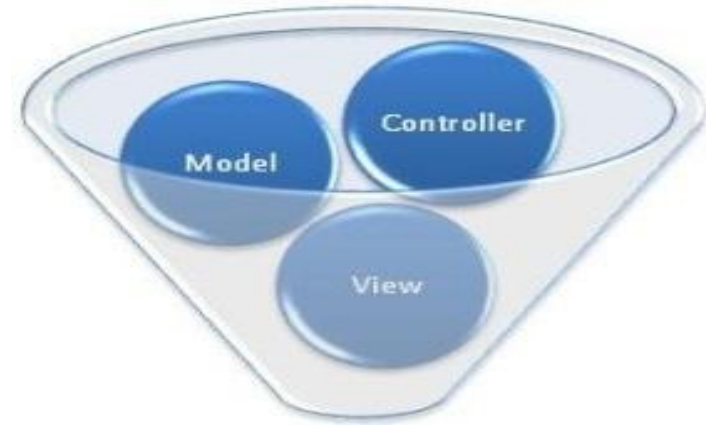


Microsoft
ASP.net MVC 5



LINQ

ThS. Nguyễn Nghiệm
0913.745.789
nghiemn@fpt.edu.vn
songlong2k@gmail.com

- ❑ Kiến trúc tổ chức của LINQ .
- ❑ Truy vấn dữ liệu

Visual C#

Visual Basic

Others

.Net Language Integrated Query (LINQ)

LINQ-enabled data sources

LINQ-enabled ADO.NET

LINQ
To Objects

LINQ
To Datasets

LINQ
To SQL

LINQ
To Entities

LINQ
To XML



Objects



Databases

```
<book>
  <title/>
  <author/>
  <price/>
</book>
```

XML

```
var result = from s in students  
             where s.Marks > 9  
             orderby s.Marks descending  
             select new { s.Name, s.Marks };
```

Biểu thức
truy vấn

Kiểu nội bộ
tự suy

Kiểu nặc danh

Khởi tạo đối tượng

```
var result2 = students  
    .Where(s => s.Marks > 9)  
    .OrderByDescending(s => s.Marks)  
    .Select(s => new { s.Name, s.Marks });
```

Biểu thức
lambda

Phương thức
mở rộng

VÍ DỤ 1: TRUY VẤN CÁC SỐ CHẴN

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };
```

```
var evens = from n in numbers  
            where n % 2 == 0  
            select n;
```

```
foreach(int n in numbers){  
    if(n % 2 == 0){  
        tích lũy số chẵn  
    }  
}
```

- ❑ **from**: chỉ ra phần tử được lấy từ tập hợp cần truy vấn
- ❑ **where**: chỉ ra điều kiện lọc
- ❑ **select**: chỉ ra đối tượng nhận được

VÍ DỤ 2: TRUY VẤN SỐ CHẴN

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };
```

```
var evens = from n in numbers  
            where n % 2 == 0  
            let rate = n / numbers.Sum()  
            orderby n descending  
            select new { number = n, rate = rate };
```

Đối tượng

```
foreach (var e in evens)  
{  
    int n = e.N  
}
```

- N
- Equals
- GetHashCode
- GetType
- number**
- rate
- ToString

int 'a'.number

Anonymous Types:
'a' is new { int number, int rate }


```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };

var evens = from n in numbers
            group n by n % 3 into g
            select new
            {
                Nhom = g.Key,
                Tong = g.Sum(),
                SoLuong = g.Count(),
                SoNN = g.Min(),
                SoLN = g.Max(),
                SoTB = g.Average()
            };
};
```

- ❑ Nhóm chia 3 dư 0: gồm 6, 45, 87
- ❑ Nhóm chia 3 dư 1: gồm 19, 13
- ❑ Nhóm chia 3 dư 2: gồm 23, 56, 5, 8

```
var evens = numbers
    .Where(n => n % 2 == 0)
    .Select(n => n);
```

```
var evens = numbers
    .Where(n => n % 2 == 0)
    .OrderByDescending(n => n)
    .Select(n => new
    {
        number = n,
        rate = n / numbers.Sum()
    });
```

```
var evens = numbers.GroupBy(n => n % 3)
    .Select(g => new
    {
        Nhom = g.Key,
        Tong=g.Sum(),
        SoLuong=g.Count(),
        SoNN=g.Min(),
        SoLN=g.Max(),
        SoTB=g.Average()
    });
```



```
var evens = from n in numbers
             where n % 2 == 0
             select n;
```



```
var evens = numbers
             .Where(n => n % 2 == 0)
             .Select(n => n);
```

```
var evens = from n in numbers
             where n % 2 == 0
             orderby n descending
             select new
             {
                 number = n,
                 rate = n / numbers.Sum()
             };
```



```
var evens = numbers
             .Where(n => n % 2 == 0)
             .OrderByDescending(n => n)
             .Select(n => new
             {
                 number = n,
                 rate = n / numbers.Sum()
             }));
```

```
var evens = from n in numbers
             group n by n % 3 into g
             select new
             {
                 Nhom = g.Key,
                 Tong = g.Sum(),
                 SoLuong = g.Count(),
                 SoNN = g.Min(),
                 SoLN = g.Max(),
                 SoTB = g.Average()
             };
```



```
var evens = numbers.GroupBy(n => n % 3)
                    .Select(g => new
                    {
                        Nhom = g.Key,
                        Tong=g.Sum(),
                        SoLuong=g.Count(),
                        SoNN=g.Min(),
                        SoLN=g.Max(),
                        SoTB=g.Average()
                    }));
```

Phương thức	Mô tả	Ví dụ
.Where(e=>điều kiện)	Lọc	Students.Where(s=>s.Marks > 9)
.GroupBy(e=>biểu thức)	Nhóm	Students.GroupBy(s=>s.Clazz)
.OrderBy(e=>biểu thức) .OrderByDescending(e=>biểu thức)	Sắp xếp	Students.OrderBy(s=>s.Name)
.Select(e=>đối tượng)	Chọn	Students.Select(s=>new{s.Name, s.Marks})
.Distinct()	Giữ 1 của các đối tượng giống nhau	Numbers.Distinct()

```
var studs = Students
    .Where(s=>s.Marks > 9)
    .OrderBy(s=>s.Marks)
    .Select(s=>s);
```

Phương thức	Mô tả	Ví dụ
.Take(số lượng)	Lấy các phần tử đầu	Students.Take(5)
.Skip(số lượng)	Bỏ qua các phần tử đầu	Students.Skip(3).Take(6)
.TakeWhile(e=>đ.kiện)	Lấy các phần tử đầu thỏa điều kiện	Students.TakeWhile(s=>s.Marks < 4)
.SkipWhile(e=>đ.kiện)	Bỏ qua các phần tử đầu thỏa điều kiện	Students.SkipWhile(s=>s.Marks < 0)

```
var result = db.Products  
    .Skip(10).Take(20)
```

```
var result = db.Customers
```

```
.Single(c=>c.Id=="A" && c.Password=="B")
```

Phương thức	Mô tả	Ví dụ
.Single(e=>đ.kiện)	Lấy 1 phần tử thỏa điều kiện. Ngoại lệ nếu không tìm thấy hoặc nhiều hơn một.	Students.Single(s=>s.Id=="Hoa")
.First()	Lấy phần tử đầu	Students.First()
.Last()	Lấy phần tử cuối	Students.Last()

Phương thức	Mô tả	Ví dụ
.Sum(e=>biểu thức số học)	Tính tổng	Students.Sum(s=>s.Marks)
.Count(e=>biểu thức số học)	Đếm số lượng	Students.Count(s=>s.Id)
.Min(e=>biểu thức số học)	Giá trị nhỏ nhất	Students.Min(s=>s.Marks)
.Max(e=>biểu thức số học)	Giá trị lớn nhất	Students.Max(s=>s.Marks)
.Average(e=>biểu thức số học)	Giá trị trung bình	Students.Average(s=>s.Marks)

❑ Var result = db.Products

~~✎~~.GroupBy(p=>p.Category)

~~✎~~.Select(g=>new{g.Key.Name, g.Count})

```
var items7 = db.Products.GroupBy(p => p.Category)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên loại
        Sum = g.Sum(p=>p.UnitPrice), //--tổng đơn giá hàng hóa của loại
        Count = g.Count(), //--số hàng hóa của loại
        Min = g.Min(p => p.UnitPrice), //--giá hàng hóa thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá hàng hóa cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });

var items8 = db.OrderDetails.GroupBy(d=>d.Product)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên hàng hóa
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị đã bán
        Count = g.Sum(p => p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

```
var items9 = db.OrderDetails.GroupBy(d => d.Product.Category)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên loại hàng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã bán
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

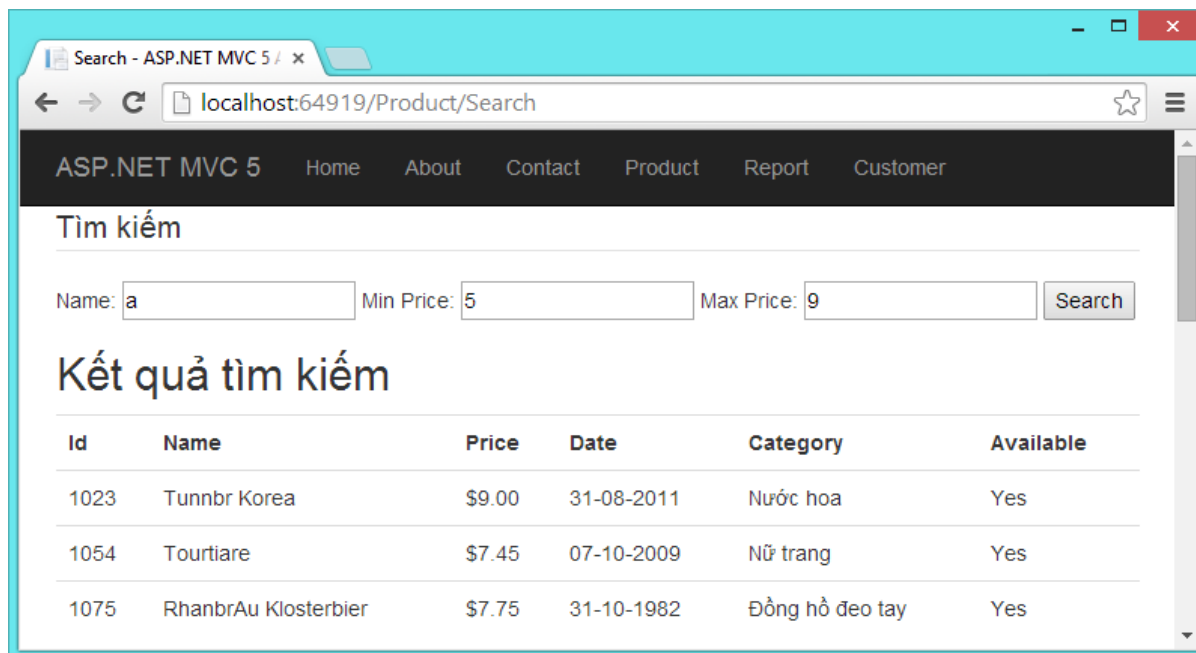
```
var items10 = db.OrderDetails.GroupBy(d => d.Order.Customer)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Fullname, //--họ và tên khách hàng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã mua
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã mua
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```



```
var items11 = db.OrderDetails.GroupBy(d => d.Order.OrderDate.Month)
    .Select(g => new ReportInfo
    {
        Group = g.Key, //--tháng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã bán
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

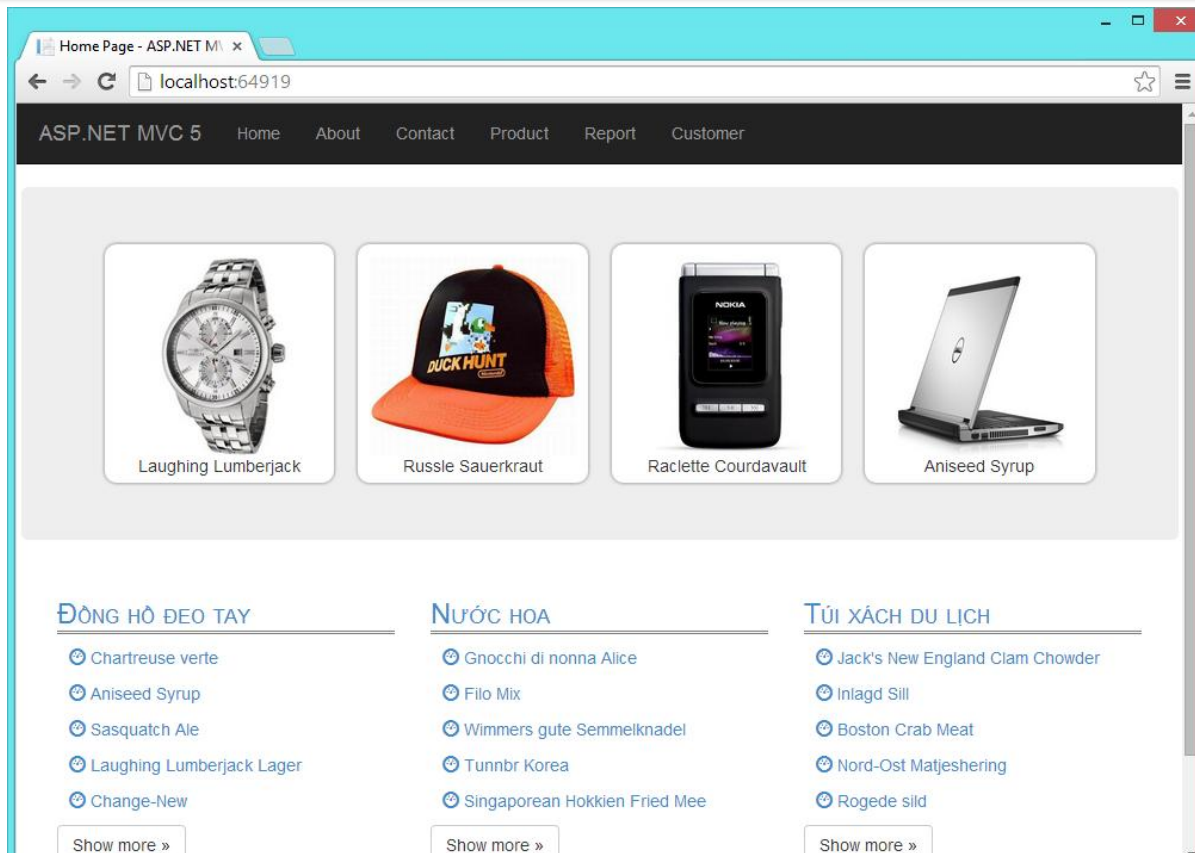
Phương thức	Mô tả	Ví dụ
.Contains(phần tử)	Tập có chứa phần tử?	Students.Contains(hoa)
.Any(e=>đ.kiện)	Ít nhất một phần tử trong tập thỏa điều kiện	Students.Any(s=>s.Marks < 3)
.All(e=>đ.kiện)	Tất cả các phần tử trong tập thỏa điều kiện	Students.All(s=>s.Marks > 5)

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };  
  
if (numbers.All(n => n % 2 == 0))  
{  
    //tất cả các số trong numbers đều là số chẵn  
}  
if (numbers.Any(n => n % 3 == 0))  
{  
    //ít nhất một số trong numbers chia hết cho 3  
}  
if (numbers.Contains(8))  
{  
    //tập numbers có chứa số 8  
}
```



Id	Name	Price	Date	Category	Available
1023	Tunnbr Korea	\$9.00	31-08-2011	Nước hoa	Yes
1054	Tourtiane	\$7.45	07-10-2009	Nữ trang	Yes
1075	RhanbrAu Klosterbier	\$7.75	31-10-1982	Đồng hồ đeo tay	Yes

```
public ActionResult Search(String Name = "",
                           double Min = double.MinValue, double Max = double.MaxValue)
{
    var list = db.Products
        .Where(p => p.Name.Contains(Name) && p.UnitPrice >= Min
                && p.UnitPrice <= Max).ToList();
    return View(list);
}
```



// truy vấn 4 sản phẩm ngẫu nhiên

```
ViewBag.Products = db.Products.OrderBy(p => Guid.NewGuid())
                        .Take(4).ToList();
```

// truy vấn 3 loại có ít nhất 5 sản phẩm ngẫu nhiên

```
var model = db.Categories
    .Where(c=>c.Products.Count > 5)
    .OrderBy(c => Guid.NewGuid()).ToList().Take(3).ToList();
```

Index - ASP.NET MVC 5 A

localhost:64919/Report

ASP.NET MVC 5
Home
About
Contact
Product
Report
Customer

Doanh số bán hàng từng loại

Loại	Doanh thu	SỐ HH	Giá TN	Giá CN	Giá TB
Máy ảnh	\$177,099.10	7906	\$7.30	\$81.00	\$22.60
Nữ trang	\$184,361.80	4204	\$5.90	\$150.00	\$45.12
Đồng hồ đeo tay	\$286,526.95	9532	\$3.60	\$263.50	\$29.24
Giày thời trang	\$105,101.20	2981	\$8.00	\$53.00	\$35.32
Điện thoại	\$211,689.70	7637	\$2.00	\$55.00	\$27.29
Nước hoa	\$91,394.40	3865	\$5.60	\$38.00	\$22.70
Máy tính xách tay	\$107,893.60	5000	\$8.00	\$43.90	\$21.40
Túi xách du lịch	\$141,623.09	7681	\$4.80	\$62.50	\$19.06